УДК 004.434

Проскуряков Кирилл Александрович

студент бакалавриата, образовательная программа «Прикладная математика и информатика», Пермский государственный национальный исследовательский университет, г. Пермь
E-mail: k.proskuryakov22@gmail.com

Лядова Людмила Николаевна

доцент кафедры математического обеспечения вычислительных систем, Пермский государственный национальный исследовательский университет, г. Пермь
E-mail: LNLyadova@gmail.com

РАЗРАБОТКА DSM-ПЛАТФОРМЫ: СРЕДСТВА ТРАНСФОРМАЦИИ МОДЕЛЕЙ ВИДА «МОДЕЛЬ-ТЕКСТ»

Proskuryakov Kirill Aleksandrovich

Student of the bachelor educational program "Applied Mathematics and Informatics" Perm State University, Perm City
E-mail: k.proskuryakov22@gmail.com

Lyadova Lyudmila Nickolaevna

Associate Professor of the Department of Computer Systems Software Perm State University, Perm City
E-mail: LNLyadova@gmail.com

DEVELOPMENT OF THE DSM PLATFORM: TOOLS FOR TRANSFORMING MODELS OF THE "MODEL-TEXT" TYPE

Аннотация: Цель исследования – апробация подхода к созданию средств генерации кода по визуальным моделям для DSM-платформы, основанной на знаниях и метамоделировании. Основа решения – многоаспектная онтология, описывающая языки, предназначенные для решения различных задач проектирования и анализа систем, и предметные области, в которых пользователи решают эти задачи. Генерация кода реализуется на основе правил трансформации, которые пользователи разрабатывают с помощью конструктора. Описаны структура онтологии и модуля трансформации, пользовательский интерфейс приложения и примеры представления языков и правил в онтологии.

Abstract: The goal of the study is to approval the approach to creating tools for generating code from visual models for a DSM platform based on knowledge and metamodeling. The basis of the solution is a multifaceted ontology, describing languages, designed to solve various tasks of designing and analysing systems, and the domains in which users solve these tasks. Code generation is implemented based on transformation rules that users develop using the constructor. The ontology structure and the transformation module structure, the application user interface and examples of languages and transformation rules representing in the ontology are described.

Ключевые слова: предметно-ориентированное моделирование, предметно-ориентированные языки, DSL, DSM-платформа, многоаспектная онтология, метамоделирование, трансформация моделей, генерация кода.

Keywords: domain specific modeling, domain specific languages, DSL, DSM platform, multifaceted ontology, metamodeling, model transformation, code generation.

Введение

Метод моделирования применяется в различных областях для решения сложных исследовательских задач, задач проектирования. При решении исследовательских задач аналитики строят множество различных моделей, соответствующих используемым методам, отражающих различные стороны моделируемых систем и процессов с различной степенью детализации. При решении задач проектирования информационных систем строится *иерархия взаимосвязанных моделей*, которые также представляют различные аспекты проектируемых систем. Эти модели строятся специалистами различных категорий. Они работают в терминах, соответствующих их области деятельности, строят определенные типы моделей, используя привычные инструменты. При этом есть опасность, что построенные модели окажутся несогласованными, будут получены противоречивые результаты.

Для решения взаимосвязанных задач проектирования и анализа сложных процессов и систем разрабатываются формальные языки, включающие средства для решения всех задач, а инструментальные средства, основанные на использовании этих языков, позволяют *поддерживать согласованность моделей* в иерархии создаваемых моделей. Одной из тенденций, поддерживаемых при создании инструментальных средств, является интеграция в одной системе множества функций анализа и проектирования, реализуемых на основе различных языков, использующих разный математический аппарат, формальные методы. «Традиционные» средства моделирования обогащаются новыми возможностями (например, язык UML – его графическая нотация – дополняется семантикой исполняемых файлов и правилами синхронизации).

С помощью UML создаются всеобъемлющие модели, не зависящие от предметной области и условий внедрения программного обеспечения. Это абстрактный инструмент мышления, который помогает в формализации знаний, является способом описания концепций, которые представляют абстрактные

решения задач разработки программного обеспечения. Исполняемый UML (*Executable UML*) стал крупным нововведением в области разработки программного обеспечения. В качестве основы для реализации архитектуры систем, управляемой моделями, Executable UML предоставляет ключевую технологию для описания предметных областей приложений независимо от платформы, но он может сделать больше, чем формализовать требования и сценарии использования, перевести их в обширный набор тестируемых диаграмм: модели на Executable UML *имеют формальную семантику действия*, так что они являются исполняемыми и тестируемыми, могут быть интерпретированы, переведены непосредственно в код компиляторами исполняемых моделей UML.

Однако универсальные языки (UML, например) не отражают потребности пользователей – специалистов в конкретных предметных областях, что затрудняет разработку моделей, провоцирует ошибки, которые являются следствием *семантического разрыва* [15], возникающего в этом случае. Привлечение различных категорий специалистов к решению задач анализа и проектирования с самых ранних этапов возможно только при наличии средств, использование которых сокращает семантический разрыв.

Существуют различные подходы к устранению семантического разрыва [1, 11]. Одним из традиционных решений является подход, основанный на *предметно-ориентированном моделировании* (*Domain Specific Modelling,* DSM), создании и использовании *предметно-ориентированных языков* (*Domain Specific Language,* DSL) – языков программирования или моделирования, специально созданных для решения определённых задач в конкретной предметной области. Такой подход называют языково-ориентированным – решение задач анализа процессов и систем или проектирования, разработки систем начинается с создания соответствующих языков. Трудоёмкость разработки смещается на создание специализированных систем программирования (или моделирования): на разработку новых языков и инструментальных средств (редакторов, препроцессоров, компиляторов или интерпретаторов для этих языков). Частично эти проблемы снимаются при использовании специального класса программного обеспечения – *DSM-платформ*, или *языковых инструментариев* [20].

Основной проблемой становится то, что разработка языков требует знаний в области формальных языков и грамматик, реализация генераторов кода – навыков использования средств трансформации моделей, языков, на которых задаются правила трансформации моделей в код. Этими знаниями и навыками специалисты в предметных областях (эксперты, аналитики) обычно не обладают.

Цель исследования – апробация подхода к созданию DSM-платформы (языкового инструментария), основанного на знаниях, который позволяет снизить трудоёмкость создания новых предметно-ориентированных языков и упростить генерацию кода по визуальным моделям, разрабатываемым пользователями. Основа решения – *многоаспектная онтология*, описывающая *языки*, предназначенные для решения различных задач проектирования и анализа систем, и *предметные области*, в которых пользователи решают эти задачи.

Анализ требований и основные принципы создания DSM-платформы, основанной на знаниях и метамоделировании, описаны в [5, 10, 14].

Трансформация моделей и генерация кода: подходы к решению задачи

Системы, в которых строятся иерархии взаимосвязанных моделей, предполагают наличие средств *трансформации моделей* при переходе к решению новых задач анализа или проектирования, при детализации построенных моделей.

В работе [16] трансформации называют «сердцем и душой» модельно-ориентированного подхода к разработке систем. Под трансформациями принято понимать и перевод моделей в другие нотации, и генерацию текстовых артефактов (исходного кода, документации, конфигурационных файлов) по визуальным моделям.

По *направлению трансформации* моделей принято делить на вертикальные и горизонтальные (рис. 1).

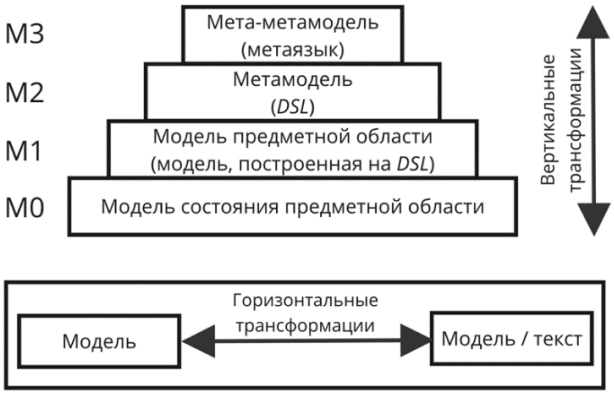

Рис. 1. Классификация трансформации моделей по направлению

Вертикальная трансформация – это преобразование модели, при котором исходная и целевая модели описаны на разных уровнях иерархии моделей. Например, трансформацией модели вышестоящего уровня в нижестоящий является создание модели с использованием языка моделирования или детализация модели, её конкретизация при создании модели состояния на основе абстрактной модели предметной области (см. рис. 1). Стоит отметить, что в современных DSM-платформах согласованность моделей при таких трансформациях поддерживается автоматически.

Горизонтальная трансформация – это трансформация, при которой исходная модель преобразуется в целевую, находящуюся на том же уровне иерархии, но описанную на другом языке, в другой нотации. Горизонтальные трансформации моделей делятся на несколько видов – трансформации вида «*Модель-Модель*» (M2M), «*Модель-Текст*» (M2T) и «*Текст-Модель*» (T2M). Однако зачастую в DSM-платформах реализуются только первые два вида горизонтальных трансформаций. Примерами трансформаций данного вида могут служить преобразование модели, описанной с помощью нотации BPMN, IDEF3 или EPC, в диаграмму активностей UML, создание SQL скрипта по модели в какой-либо нотации диаграмм «сущность-связь» (Entity-Relationship Diagram, ERD) или по диаграмме классов, а также генерация кода на языке программирования по блок-схеме алгоритма.

В статьях [6, 8] предлагается делить трансформации такого вида на две

группы на основании используемого в них подхода: трансформации, основанные на посетителях (*visitor-based*) и трансформации, основанные на шаблонах (*template-based*) или паттернах (*pattern-based*).

Подход к трансформациям моделей, основанный на шаблонах (паттернах), используется чаще [7, 12]. Согласно исследованию [12], одними из наиболее часто используемых средств трансформаций, основанных на шаблонах, являются Acceleo и Xpand, а также Velocity и T4. Этот же подход был реализован и в DSM-платформе MetaLanguage [17, 18].

Шаблон состоит из двух частей – статической и динамической. *Статическая* часть является общей для всех моделей, а *динамическая* использует информацию, «извлечённую» из полученной на вход модели. Для данного подхода консорциумом OMG был разработан стандарт MOF *Model to Text Transformation Language* (MTTL), являющийся частью MDA-подхода (URL: https://www.omg.org/spec/MOFM2T/1.0/PDF). Стандарт MTTL применяется для трансформаций MOF-совместимых моделей.

Оценка и сравнение средств создания DSL в различных DSM-платформах приведены в [9, 19]. Однако ни один из существующих промышленных языковых инструментариев не имеет средств автоматизации разработки языков DSL, а средства трансформации в существующих DSM-платформах реализуются на основе использования *специальных языков описания трансформаций*, которые сложны для изучения и использования (табл. 1). Используются, например, скриптовый язык VTL, где динамическая часть правил описывается языками C# или Visual Basic, язык MTTL, где для описания сложной логики пользуется язык Java, и пр. Применение этих языков требует профессиональных навыков: разработчик должен владеть и языками описания трансформаций и языками программирования, быть экспертом в предметных областях, для которых разрабатываются модели на визуальных DSL, и владеть профессиональными знаниями и навыками в области формальных языков и грамматик.

Таблица 1. Результаты сравнения средств трансформации DSM-платформ

Название средства	**Язык описания шаблона**	**Способ задания правила**	**Полнота языка описания шаблона**
Acceleo	MTTL	Текстовый	+
Xpand	Собственный, Java	Текстовый	+
Velocity	VTL	Текстовый	+
T4	C#, VB	Текстовый	+
Metalanguage	Собственный	Визуальный, Текстовый	–

DSM-платформа, управляемая знаниями: подход к разработке

Процесс разработки *правил трансформации* можно упростить, если использовать визуальные средства для их построения (конструкторы, построители правил), а не текстовые языки описания трансформаций. Этот подход прошёл апробацию при создании языкового инструментария MetaLanguage. Предлагается реализовать подобные средства трансформации моделей на основе онтологии [5].

Функциональные требования к средствам моделирования реализуются соответствующими функциональными блоками системы (DSM-платформы).

Подсистема *создания языков* позволяет пользователям создавать DSL, описывая их метамодели, или же определяя правила автоматической генерации метамоделей – правила отображения моделей предметных областей на существующие языки (таким образом созданные языки настраиваются на специфику предметной области) [2, 3].

Подсистема *работы с визуальными моделями* включает *редактор* визуальных моделей, *средства трансформации* моделей и генерации текстовых артефактов (средства трансформации «Модель-Текст») на основе правил, определяемых пользователями. *Модуль импорта и экспорта* моделей позволяет сохранять и загружать модели в заданных форматах. Управление моделями осуществляется через *браузер моделей*. Один и тот же *редактор* используется как для разработки метамоделей визуальных языков, так и для разработки моделей. Редактор настраивается на использование нового языка по его описанию в онтологии (метамодели языка).

Ядром системы является *многоаспектная онтология*, которая хранит в себе все необходимые для функционирования системы знания. Для работы с онтологией используется *браузер онтологий*. В систему могут быть загружены онтологии, созданные с помощью сторонних редакторов. Разработанные в системе онтологии могут быть выгружены для использования в других приложениях.

Онтология визуальных языков содержит метамодели визуальных языков, описанных на метаязыке HPGPR, разработанном на основе языка GOPPRR, расширенного описанием гиперрёбер, определённых в модели гиперграфов с полюсами (*HP*-графов) [13]. *Онтология текстовых языков* содержит описание грамматик языков (основа описания – синтаксические диаграммы Вирта). Языки классифицируются по их назначению (по решаемым с их помощью задачам).

Онтология предметных областей включает описания (модели) предметных областей, для которых разрабатываются языки и модели на описанных языках для решения задач пользователей.

Онтология *правил проецирования* содержит правила для генерации новых DSL, основанные на сопоставлении элементов метамоделей языков с понятиями онтологий предметных областей и связей между ними. Правила отображения задают семантику элементов языка для конкретной предметной области.

Онтология моделей обеспечивает хранение и поиск визуальных моделей, разработанных с использованием созданных языков, метамодели которых представлены в онтологии языков. Связь моделей с языками, использованными для их создания, позволяет контролировать выполнение операций над моделями, их соответствие заданным в метамоделях ограничениям, а также выполнять трансформации моделей по правилам, определённым для соответствующих языков.

Онтология правил трансформации ставит в соответствие элементам визуальных языков конструкции текстовых языков, заданные в грамматиках этих языков, представленных в соответствующей онтологии. В данном случае не

рассматриваются трансформации типа «Модель-Модель» – решается только задача генерации кода по визуальной модели – трансформации «Модель-Текст».

Онтологии «соединяются» в многоаспектную онтологию с помощью блоков F_n, которые связывают описанные онтологии при реализации функций системы:

– F_1 – формирование правил проецирования (отображения) онтологии предметной области на метамодель визуального языка для автоматической генерации нового языка, предназначенного для решения тех же задач, что и исходный язык, но с учётом специфики выбранной предметной области;

– F_2 – генерация визуальных языков на основе правил проецирования (отображения онтологии предметной области на метамодели визуальных языков);

– F_3 – определение правил трансформации типа «Модель-Текст»;

– F_4 – применение правил трансформации и связь между исходной визуальной моделью и кодом, сгенерированным на её основе (для поддержания согласованности модели и кода).

Для описания онтологии используется язык OWL, для последующих запросов к ней при генерации языков и выполнении трансформаций – язык SPARQL.

Разработка средств генерации кода

Правила генерации кода (трансформации «Модель-Текст») создаются на основе метамодели визуального языка, включённой в онтологию, и правил грамматики текстового языка, представленным в виде диаграмм Вирта. С помощью конструктора правил задаётся соответствие между элементами метамодели исходного языка и элементами грамматики целевого языка (в качестве примера приведено представление синтаксического правила для оператора создания таблицы CREATE TABLE – показано на рис. 2). Правила соответствия между описаниями языков также представляются в онтологии. Между различными элементами метамоделей и элементами синтаксических диаграмм устанавливаются различные типы отношений. Пользовательский интерфейс конструктора правил трансформации показан на рис. 3

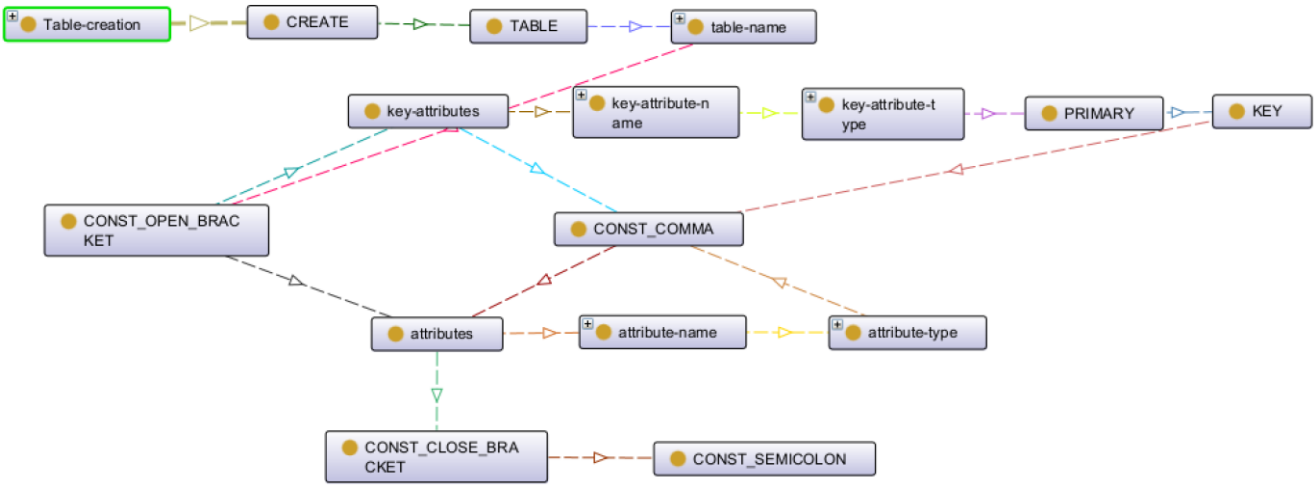

Рис. 2. Представление синтаксической диаграммы в онтологии

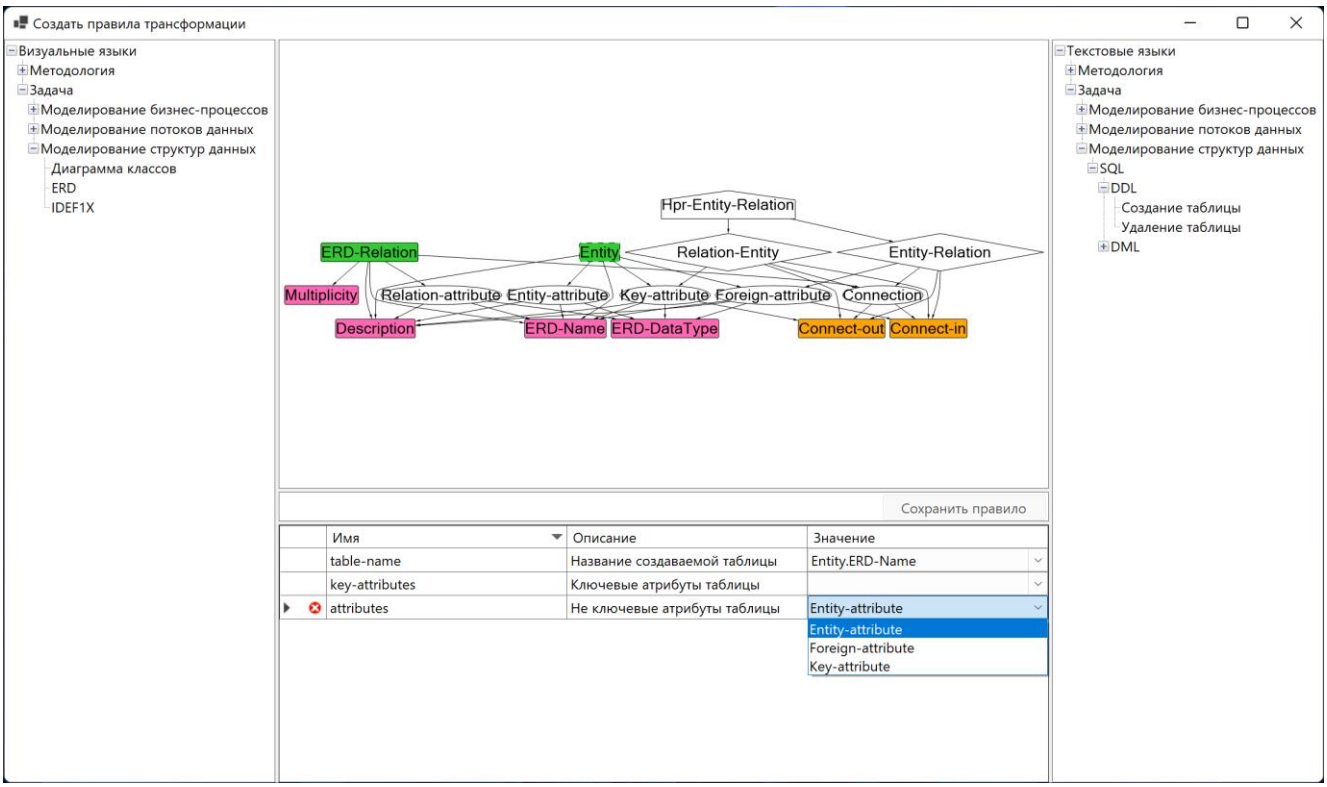

Рис. 3. Форма конструктора правил трансформации типа «Модель-Текст»

Для добавления правил, их редактирования и интерпретации при генерации кода разработаны соответствующие алгоритмы. Пример (фрагмент онтологии, представляющий правило трансформации модели в нотации ERD в код на языке DDL (SQL)) приведён на рис. 4.

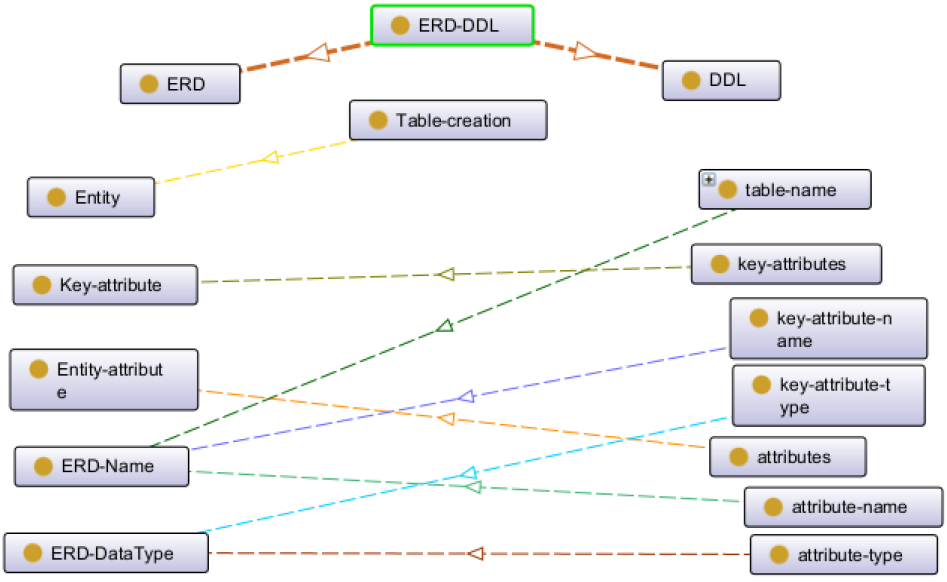

Рис. 4. Отношения в правилах трансформаций в онтологии

Структура модуля трансформации моделей показана на рис. 5. Алгоритмы разработки правил трансформации реализованы в классе *TransformationRulesBuilder*. Для выполнения SPARQL-запросов, необходимых для реализации трансформаций, используется класс *TransformationsQueryExecutor*. Код на SQL, сгенерированный в соответствии с заданными правилами, можно сохранить в файле и выполнить для создания БД.

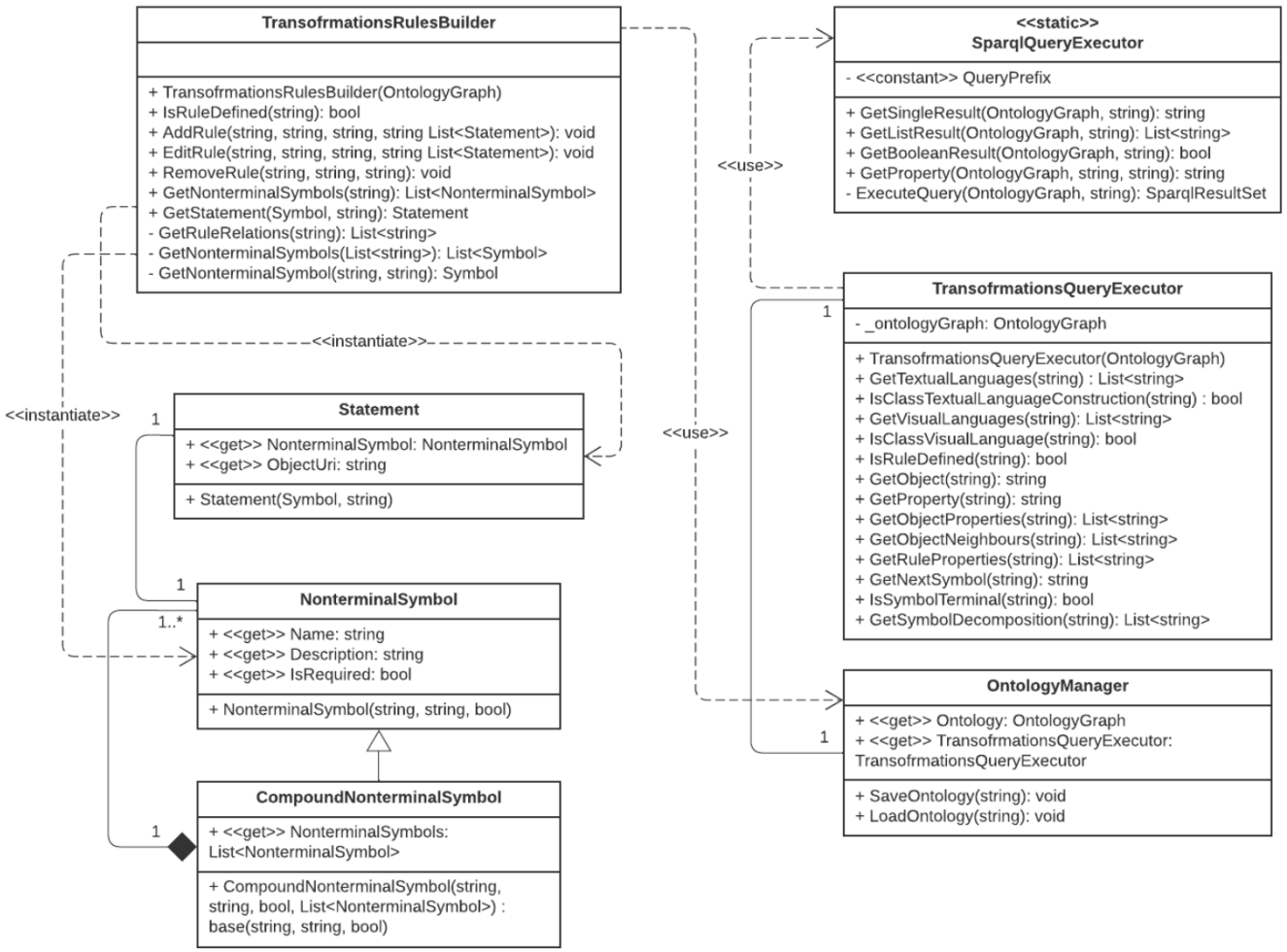

Рис. 5. Диаграмма классов модуля трансформаций

Заключение

Разработанный прототип показал практическую значимость описанного подхода к созданию средств трансформации моделей. Разработанные алгоритмы могут быть усовершенствованы, чтобы снять ограничения на вид правил генерации кода, расширить их возможностями определения альтернативных вариантов генерации кода, циклами для обработки конструкций, допускающих наличие произвольного числа элементов и т. п. Кроме того, предполагается реализовать средства доопределения правил на основе знаний, представленных в онтологии. Показано, как эти средства могут быть интегрированы в состав аналитических платформ для решения исследовательских задач [4, 13, 21].

Список литературы

1. Дегтярев А. А. Анализ способов сокращения семантического разрыва при разработке программного обеспечения / А. А. Дегтярев // Технологии разработки информационных систем: материалы конференции ТРИС-2012. Т. 1. – Таганрог: Изд-во ТТИ ЮФУ, 2012. – С. 146–150.

2. Ермаков И. Д. Автоматизация разработки предметно-ориентированных языков на основе онтологии / И. Д. Ермаков // Технологии разработки информационных систем: сб. матер. XII Международной научно-технической конференции ТРИС-2022. – Таганрог: Издательство ЮФУ, 2022. С. 63–71.

3. Кулагин Г. А. Разработка иерархии предметно-ориентированных языков описания алгоритмов для устройств компании GalileoSky / Г. А. Кулагин // Технологии разработки информационных систем: сб. матер. XII Международной научно-технической конференции ТРИС-2022. – Таганрог: Издательство ЮФУ, 2022. С. 80–88.

4. Лядова Л. Н. О подходе к разработке аналитической платформы, основанной на знаниях и метамоделировании / Л. Н. Лядова, В. С. Заякин, Н. М. Суворов // Информатизация и связь. 2022. № 5. С. 85–90.

5. Лядова Л. Н. Автоматизация разработки предметно-ориентированных языков на основе многоаспектных онтологий / Л. Н. Лядова // Информатизация и связь. 2021. № 8. С. 48–52.

6. Czarnecki K. Feature-based survey of model transformation approaches / K. Czarnecki, S. Helsen // IBM Systems Journal. 2006. V. 45, No. 3. P. 621–645.

7. Ding J. Code Generation Approach Supporting Complex System Modeling based on Graph Pattern Matching / J. Ding, J. Lu, G. Wang, J. Ma, D. Kiritsis, Y. Yan // IFAC-PapersOnLine. – 2022. – V. 55, Issue 10, – P. 3004–3009. DOI: 10.1016/j.ifacol.2022.10.189.

8. Kahani N. Survey and classification of model transformation tools / N. Kahani, M. Bagherzadeh, J. Cordy // Software & Systems Modeling. – 2019. – V. 18. – P. 2361–2397. DOI: 10.1007/s10270-018-0665-6.

9. Kelly S. Empirical Comparison of Language Workbenches / S. Kelly // DSM’13: Proceedings of the 2013 ACM workshop on Domain-specific modeling. Indianapolis. – 2013. – P. 33–38. DOI: 10.1145/2541928.2541935.

10. Kulagin G. Ontology-Based Development of Domain-Specific Languages via Customizing Base Language / G. Kulagin, I. Ermakov, L. Lyadova // Proc. IEEE 16th International Conference on Application of Information and Communication Technologies (AICT). – Washington : IEEE. – 2022. – P. 1–6. DOI: 10.1109/AICT55583.2022.10013619.

11. Lapshin V. The conception of a method of analyzing business activities as part of the process of developing information systems / V. Lapshin, O. Shevchenko, K. Kandyba // International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, SGEM. – 2020. – V. 20. No. 2.1. – P. 349–356. DOI: 10.5593/sgem2020/2.1/s07.045.

12. Luhunu L. Comparison of the expressiveness and performance of template-based code generation tools / L. Luhunu, E. Syriani // Proc. ACM SIGPLAN 10th International Conference on Software Language Engineering (SLE 2017), New York. – 2017. – P. 206–216. DOI: 10.1145/3136014.3136021.

13. Lyadova L. An Ontological Approach to the Development of Analytical Platform Language Toolkits / L. Lyadova, N. Suvorov, V. Zayakin, E. Zamyatina // Proc. IEEE 16th International Conference on Application of Information and Communication Technologies (AICT). – Washington : IEEE. – 2022. – P. 1–6. DOI: 10.1109/AICT55583.2022.10013576.

14. Lyadova L. An Ontology-Based Approach to the Domain Specific Languages Design / L. Lyadova, A. Sukhov, M. Nureev // Proceedings of the 15th International Conference on Application of Information and Communication Technologies (AICT2021). Baku, Azerbaijan: IEEE. – 2021. – P. 1–6. DOI: 10.1109/AICT52784.2021.9620493.

15. Robert F. Model-driven Development of Complex Software: A Research Roadmap / F. Robert, R. Bernhard R. // Proceedings IEEE Transactions on Software Engineering. Minneapolis. – 2007. – P. 37–54. DOI: 10.1109/FOSE.2007.14.

16. Sendall S. Model Transformation: The Heart and Soul of Model-Driven Software Development / S. Sendall, W. Kozaczynski // IEEE Software. – 2003. – V. 20. Issue 5. – P. 42–45. DOI: 10.1109/MS.2003.1231150.

17. Sukhov A. O., Lyadova L.N. Horizontal Transformations of Visual Models in MetaLanguage System / A. O. Sukhov, L. N. Lyadova // Proceedings of the 7th Spring/Summer Young Researchers' Colloquium on Software Engineering, SYRCoSE 2013. – Kazan : -, 2013. P. 41–46.

18. Sukhov A. O. Visual Models Transformation in MetaLanguage System / A. O. Sukhov, L. N. Lyadova // Advances in Information Science and Applications. Volumes I & II. Proceedings of the 18th International Conference on Computers (part of CSCC '14). – Vol. 1–2. Santorini Island : CSCC, 2014. P. 460–467.

19. Tolvanen J.-P. Effort used to create domain-specific modeling languages / J.-P. Tolvanen, S. Kelly // MODELS' 18: ACM/IEEE 21th International Conference on Model Driven Engineering Languages and Systems. – 2018. – P. 235–244.

20. Tolvanen, J.-P. Model-driven development challenges and solutions – experiences with domain-specific modeling in industry / J.-P. Tolvanen, S. Kelly // Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development, Rome, Italy. – 2016. – P. 711–719.

21. Zayakin V. S. An Ontology-Driven Approach to the Analytical Platform Development for Data-Intensive Domains / V. S. Zayakin, L. N. Lyadova, V. V. Lanin, E. B. Zamyatina, E. A. Rabchevskiy // Knowledge Discovery, Knowledge Engineering and Knowledge Management: 13th International Joint Conference, IC3K 2021, Revised Selected Papers. – 2023. – V. 1718. – P. 129–149. DOI: 10.1007/978-3-031-35924-8_8.